Journal of Korea Multimedia Society Vol. 20, No. 5, May 2017(pp. 835-843)
https://doi.org/10.9717/kmms.2017.20.5.835

게임 어플리케이션을 위한 컨볼루션 신경망 기반의 실시간 제스처 인식 연구

채지훈[†], 임종헌[††], 김해성[†††], 이준재[†††]

Study on Real-time Gesture Recognition based on Convolutional Neural Network for Game Applications

Ji Hun Chae[†], Jong Heon Lim[†], Hae Sung Kim[††], Joon Jae Lee[††]

ABSTRACT

Humans have often been used gesture to communicate with each other. The communication between computer and person was also not different. To interact with a computer, we command with gesture, keyboard, mouse and extra devices. Especially, the gesture is very useful in many environments such as gaming and VR(Virtual Reality), which requires high specification and rendering time. In this paper, we propose a gesture recognition method based on CNN model to apply to gaming and real-time applications. Deep learning for gesture recognition is processed in a separated server and the preprocessing for data acquisition is done a client PC. The experimental results show that the proposed method is in accuracy higher than the conventional method in game environment.

Key words: Convolutional Neural Network, Gesture Recognition, Game Applications

1. 서 론

인간은 서로간의 의사소통을 위해 언어 외에도, 손짓, 몸동작을 통해 대화를 진행한다. 컴퓨터와 사람 사이의 의사소통 역시 이와 다르지 않다. 마우스나 키보드 등 외부 입력기기를 이용하여 컴퓨터에게 명령을 내리고, 키넥트와 같은 3D 센서를 활용하여 제스처를 입력한다. 스마트폰, 태블릿의 경우 터치스크린을 활용하여 싱글, 멀티터치를 기반으로 2D 공간에서 제스처를 입력하여 인식하고 있으며 가상현실(VR) 혹은 증강현실(AR)의 경우, 별도의 무선 컨트롤러나 3D 센서를 활용하여 3D 공간에서 제스처를 인식한다[1].

제스처 인식은 다양한 플랫폼에서 활용되어 왔으며 특히, 가상현실에서 필요성이 대두되고 있다. 가상현실의 경우 기존의 입력장치가 불편하고 직관성이 떨어짐에 따라 제스처 인식에 대한 필요성이 확대되고 있다. 또한 게임 어플리케이션에서 제스처 인식을 기반으로 폭 넓게 활용함으로서, 게임 콘텐츠뿐만 아니라 인터페이스 조작 등 게임에서 요구하는 동작을 제스처를 통해서 입력할 수 있다. 하지만 게임 어플리케이션에서 제스처 인식을 수행하는데 어려움이 따른다. 예를 들어 제스처 인식 기반의 게임은 실시간으로 매 프레임마다 렌더링(Rendering) 과정을 거치는 것과 별도로 제스처 인식 알고리즘을 처리한다. 알고리즘의 비용이 높을 경우, FPS(Frame Per

※ Corresponding Author : Joon Jae Lee, Address: 1095 Dalgubeol-daero, Dalseo-gu, Daegu 42601, Korea, TEL : +82-53-580-6682, FAX : +82-53-580-5165, E-mail : joonlee@kmu.ac.kr
Receipt date : Apr. 5, 2017, Revision date : Apr. 21, 2017
Approval date : Apr. 21, 2017

[†] Dept. of Computer Engineering, Graduate School, Keimyung University
(E-mail : cowlgns21@naver.com,stjylim@naver.com)
[††] Faculty of Computer Engineering, Keimyung University
(E-mail : gotjd3697@naver.com,joonlee@kmu.ac.kr)

Second)가 크게 감소하는 현상이 발생한다. 이러한 예는 사용자가 게임을 이용할 때, 몰입감이 떨어지는 원인이 된다. 이와 같이 게임 어플리케이션에서 실시간으로 제스처를 정확하게 인식하는 것은 난이도를 요구하며 특히 가상현실 구현에 필요한 높은 하드웨어 성능으로 인해 제스처 인식에 많은 자원을 할당하기 쉽지 않다.

최근 컴퓨터비전, 기계학습 분야에서는 딥 러닝 알고리즘을 활용하여 다양한 문제들을 해결하고 있다. 특히 CNN(Convolutional Neural Network), RNN (Recurrent Neural Network)와 같은 모델의 경우 이미지 인식 및 분류, 음성인식, 자연어 처리 등 다양한 분야에서 높은 성능을 보이며 활용범위가 넓어지고 있다[2].

본 논문에서는 앞에서 소개한 딥 러닝 알고리즘 중 하나인 CNN 통해서 제스처 인식을 수행한다. 또한 본 논문에서 가상현실 공간, 컴퓨터 등 다양한 환경에서 제스처를 인식하고, 게임 어플리케이션 환경에서 실시간으로 제스처 인식이 가능한 것을 목표로 한다.

본 논문의 구성은 다음과 같다. 2장에서는 2D 제스처를 인식하는 기존 연구 사례를 설명한다. 3장에서 CNN 모델이 학습할 2D 제스처 데이터 구성 및 전처리 방법 대해서 설명한다. 그리고 4장에서 CNN 모델의 핵심 구조를 설명한다. 5장에서는 실시간으로 게임 어플리케이션을 처리하기 위해서, 딥 러닝 연산을 전담하는 서버와 게임 어플리케이션을 처리하는 클라이언트 간의 네트워크 구조를 설명한다. 6장에서는 제안한 CNN 모델과 전처리 과정을 포함한 제스처의 인식 속도 실험을 진행한다. 그리고 7장은 CNN 모델의 분류 실험을 진행한다. 8장에서는 실제 게임 어플리케이션에 적용한 결과를 보여준다. 마지막 9장에서는 결론을 맺으며 향후 연구 방향을 제시한다.

2. 기존 연구

2D 제스처를 인식하는 방법에 대한 연구는 다양한 곳에서 지속적으로 진행되어 왔다. 특히, 게임 인터페이스에서 K-nearest neighbor 알고리즘을 이용하여 보다 효과적으로 적은 연산량을 통해 제스처를 인식할 수 있는 모델을 찾아볼 수 있다. 이처럼 기존 연구에서 제스처를 인식한 모델을 게임 인터페이스 환경에서 적용한 것은 Fig. 1과 같다. 하지만 비슷한 모양을 가진 제스처를 분류하거나 실사용 측면에 있어 인식률이 떨어지는 단점이 있었다[3].

Fig. 1. Gesture recognition for game interface[3].

2D 제스처를 입력하고 인식하는데 있어 다양한 분류의 데이터를 정확하게 예측하기 위해서는 이동 회전, 크기 등에 강건하고 특징을 잘 추출할 수 있는 특징자와 분류기가 필요하다. 딥 러닝의 모델 중 하나인 CNN의 경우 이 같은 특징을 효과적으로 추출하고 강건한 분류를 할 수 있다. 특히 회전의 경우 다양한 분포를 가진 대량의 데이터를 학습하면 적절한 대응이 가능한 것으로 알려져 있다[4]. 또 다른 연구사례의 경우 립 모션(leap motion)을 통해 제스처를 입력하여 9600개의 데이터를 지도학습 방식으로 학습하고 12종류의 제스처를 분류할 때 92.4%의 인식률을 확인할 수 있다[5]. 하지만 게임 어플리케이션과 같이 실시간으로 처리하기 위해서는 연산속도나 CPU 혹은 GPU의 남은 자원 등 다양한 문제가 있는 것이 사실이다.

3. 2D 제스처 데이터 구성 및 전처리 방법

3.1 2D 제스처 데이터 구성

CNN 모델 학습을 위한 2D 제스처 데이터를 구성하기 위해 10종류의 제스처 종류를 결정하였고 패턴은 Fig. 2와 같다. 학습 데이터 9204개와 테스트 데이터 1000개를 구성하였으며 데이터를 수집하기 위해 4명의 실험 군이 직접 손으로 입력하여 패턴을 이미지로 저장하였다.

데이터 수집 환경은 유니티 5를 활용해서 Fig. 3과 같이 GUI(Graphic User Interface)를 구축했다. 좌측 상단에는 수행해야 할 항목을 표시했다. 첫 번째는 이미지 크기, 두 번째는 이미지의 픽셀 누락 현상을

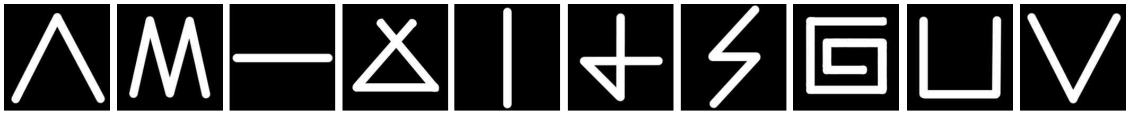


Fig. 2. Gesture type.

Fig. 3. GUI for data collection.

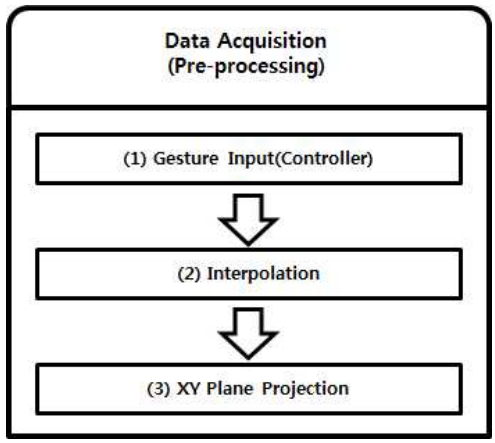


Fig. 4. Data acquisition process.

보완하기 위해서 보간 크기를 정한다. 자세한 사항은 3.2에서 언급한다. 세 번째는 저장 할 패턴 종류 숫자를 키보드의 숫자 패드를 통해서 입력 한다. 네 번째로 데이터를 입력한다. 하지만 데이터는 환경에 따라 입력하는 수단이 달라진다. 본 논문의 인터페이스는 PC, VR(Virtual Reality)를 지원한다. PC는 마우스를 통해서 왼쪽 버튼을 누른 후 드래그를 통해서 제스처를 입력하는 방식을 취한다. 하지만 VR은 독자적인 컨트롤러에서 나오는 직선을 활용해서 제스처를 입력한다. 마지막으로 마우스의 오른쪽 버튼을 통해서 입력한 제스처 데이터를 날짜와 시간 순으로 저장한다.

3.2 2D 제스처 데이터 전처리 방법

이미지의 크기는 32x32이며 제스처 패턴 입력 및 수집 과정에서 전처리 과정을 Fig. 4와 같이 수행한다. Fig. 4의 (1)에서는 PC 및 가상현실(VR) 환경의 컨트롤러를 통해서 3차원의 벡터로 구성 된 제스처 데이터를 획득한다. Fig. 4의 (2)과정에서 제스처를 입력할 때 사용자가 데이터를 입력하는 시간이 매 업데이트마다 데이터를 처리하는 시간보다 짧을 경우, 데이터를 이미지로 변환할 때 일부 픽셀이 누락되는 현상이 발생한다. 이러한 현상을 줄이기 위해서 보간 함수를 수행한다. 보간 함수는 식 (1)과 같이 정의된다[3].

$$t=[0,1],\ v(t)=(1-t)v(p)+tv(c) \tag{1}$$

선형 보간법(linear interpolation)은 두 벡터 사이의 위치한 값을 추정하기 위해서 선형적으로 보간하는 방법이다. 식 (1)에서 t는 보간 범위로 0-1 사이의 값이다. 이전 업데이트에 입력한 벡터 $v(p)$, 현재 업데이트에서 입력한 벡터 $v(c)$를 통해서 새로운 벡터 $v(t)$를 획득 할 수 있다. 반복 단위를 0.1 또는 0.01로 설정한 후, 매 업데이트 마다 t값이 1이 될 때까지 보간 함수를 반복 수행한다.

Fig. 4의 (3) 과정에서는 획득한 데이터는 XY축의 평면에 투영함으로서 이미지로 변환한다.

첫 번째로, 획득한 벡터들 모두 포함 할 수 있는 정사각형을 생성하기 위해서, 이미지 너비가 될 수 있는 값을 산출한다. 식 (2)와 같이 정의 할 수 있다.

이미지 너비가 될 수 있는 값을 산출하기 위해서, 입력 받은 벡터들 중 가장 좌측 상단 벡터의 x값을 x_{tl}, 입력 받은 벡터들 중 가장 좌측 상단 벡터의 y값을 y_{tl}, 입력 받은 벡터들 중 가장 우측 하단 벡터의 x값을 x_{br}, 입력 받은 벡터들 중 가장 우측 하단 벡터의 y값을 y_{br}로 정의한다. 그리고 이를 통해서 결정된 이미지 너비를 $Max\,Distance$라고 정의한다. 앞서 지정

한 값을 통해서 $Max\ Distance$의 후보는 x값을 통해서 산출된 거리 값 D_x와 y값을 통해서 산출된 거리 값 D_y를 선정 할 수 있으며, 식 (2)에 따라 이미지 너비가 결정된다.

$$\begin{aligned} &D_x = abs(x_{tl} - x_{br}),\ D_y = abs(y_{tl} - y_{br}) \\ &(D_x = D_y) \\ &Max\ Distance = D_x, \\ &(D_x > D_y) \\ &Max\ Distance = D_x, \\ &(D_x < D_y) \\ &Max\ Distance = D_y \end{aligned} \tag{2}$$

두 번째로, 획득한 이미지 너비를 통해서 픽셀의 한 개의 너비를 얻는다. 픽셀 너비를 구하는 식 (3)으로 정의한다.

$$CellSize = Max\ Distance / GridSize \tag{3}$$

식 (3)에서 요구하는 $GridSize$는 이미지의 실제 너비가 되기 때문에, 사용자가 미리 설정한다. 본 논문에서는 크기 32×32 이미지를 사용함에 따라 $GridSize$를 32로 설정한다. 이후, 결정된 픽셀의 너비 및 높이인 $CellSize$를 통해서 3차원 벡터를 2차원의 이미지 픽셀 위치로 변환 할 수 있다.

$$\begin{aligned} X &= ((x_{p_k} - x_{tl}) / CellSize) \\ Y &= ((y_{p_k} - y_{tl}) / CellSize) \end{aligned} \tag{4}$$

세 번째로, 상기 결정된 $CellSize$를 통해서 픽셀의 위치 값을 결정한다. 픽셀의 위치가 픽셀 위치를 도출하는 것은 식 (4)와 같이 정의한다.

입력 받은 벡터 중에서 픽셀이 될 수 있는 모든 벡터들을 p_k라고 할 때, x값을 x_{p_k}, y값을 y_{p_k}라고 표현한다. 이미지의 중심점을 설정하기 위해서, 입력 받은 벡터들 중 가장 좌측 상단 벡터의 x값을 x_{tl}, 입력 받은 벡터들 중 가장 좌측 상단 벡터의 y값을 y_{tl}로 정의한다. 입력 받은 3차원 벡터가 2차원의 픽셀 위치로 변환함으로서, 각 픽셀 위치마다 색상을 지정한다. 이를 통해서 크기 32×32 이미지를 생성할 수 있다.

Fig. 5. Data interpolation comparison.

3.3 데이터 전처리 결과

Fig. 5는 보간 함수를 적용 했을 때 (a)와 적용한 후 (b)를 비교한 그림이다. Fig. 5의 (a)의 직선을 보면 픽셀이 손실된 현상을 볼 수 있다. 이는 제스처를 입력하는 과정에서 업데이트 속도에 의해 중간 데이터가 손실되는 것을 원인으로 볼 수 있으며 이 구간을 선형 보간을 통해 제스처 패턴을 매끄럽게 변활될 수 있도록 수행한다. 최종적으로 (b)와 같이 데이터의 손실을 최소화하고 정상적인 패턴을 얻을 수 있다.

Fig. 4의 마지막 (3) 과정에서는 보간 된 데이터를 2D 평면에 투영시켜 이미지화 하는 과정을 진행한다. 제스처 모양의 너비를 계산하여 설정된 이미지 크기만큼 한 픽셀의 크기를 결정하고, 연속적인 벡터의 연속적인 위치 값을 픽셀의 위치로 양자화한다. 전처리 과정을 거쳐 나온 최종 제스처의 이미지는 Fig. 6과 같다.

Fig. 6을 살펴보면 이미지 크기 32×32에 맞춰 양자화 된 최종 이미지를 확인할 수 있다. 2번과 4번의 경우 각각 가로, 세로 패턴에 해당되며 해당 방향으로 긴 형태를 가지고 있기 때문에 좌측상단으로 몰려있는 현상을 볼 수 있다. 패턴의 중심에 맞춰서 이미지를 생성할 경우 추가적인 전처리 연산이 필요하기 때문에 이 같은 과정은 생략하였으며 이 문제가 인식률에 큰 영향을 미치지 않았다.

4. 제스처 인식을 위한 딥 신경망 모델

4.1 Convolutional Neural Network 구조

일반적으로 제스처 패턴의 경우 사용자의 그리는 방식이나 스타일, 왼손 혹은 오른손 사용 여부 등에

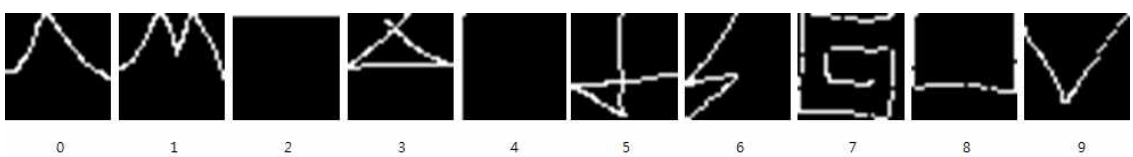


Fig. 6. Result of gesture data using pre-processing.

따라 다양한 모양을 나타낸다. 이는 제스처를 인식하는데 있어 복잡도를 심화시킨다. 하지만 CNN 모델은 대량의 데이터를 학습한 후 추론하는데 있어 좋은 성능을 가지고 있다. 특히 이동(translation), 왜곡(skew), 크기(scale)와 같은 변환에 대해 강건한 모습을 보여주며 결과적으로 제스처를 인식하는데 있어 높은 인식률을 확보할 수 있다. 본 연구와 같은 문제의 경우 사용자에 의한 모양의 다양성 및 변환에 대한 부분 등에서 필기체를 인식하는 것과 유사한 점이 많다. 따라서 MNIST와 같은 필기체 인식 모델을 제스처를 인식하는데 활용할 수 있을 것으로 예상하였다.

필기체 인식을 위한 CNN 모델 중 LeNet[6] 모델을 차용하여 2D 제스처에 대한 분류를 시도하였다. 모델의 구조는 Fig 7과 같다. 기울기 사라짐(gradient vanishing)문제를 해결하기 위해 ReLU(Rectified Linear Unit)를 활성함수로 사용하였으며 학습데이터가 적어 실험 과정에서 과적합(over fitting) 현상이 심하게 발생하여 Dropout 기법을 활용했다[7]. 총 10개의 제스처를 분류하기 위해 10개의 출력을 사용했고, Dropout 기법은 첫 번째와 두 번째 fully-Connected 레이어 사이에 적용했다. 또한 이미지 데이터는 32x32 크기를 가지지만 제시한 모델의 연산 시간을 줄이기 위해서 입력 시 28x28로 이미지 크기를 조절했다.

4.2 네트워크 구조

최근 딥 러닝은 빠른 학습과 분류를 위해 GPU의 병렬처리를 활용한다. 이 연산은 기존의 어플리케이션에는 적용이 가능하지만, 특히 게임, 가상현실과 같이 GPU를 실시간으로 사용하는 어플리케이션의 경우 큰 부담이 된다. 최근 가상현실(VR)과 같은 경우 권장사양으로 최신 그래픽카드를 필요로 하는 만큼 하나의 PC에서 상호 실행이 사실상 불가능 한 경우가 많다.

이 같은 문제를 해결하기 위해 본 논문에서는 네트워크 통신을 통해 딥 러닝이 연산하는 부분을 서버에서 전담하여 처리할 수 있도록 하였으며, 클라이언트에 해당하는 어플리케이션은 서버로 제스처 데이터를 보내고 인식 결과만 받을 수 있도록 Fig. 8과 같이 구성하였다. 패킷의 안정성을 위해 TCP 통신을 사용하였으며 패킷에 대한 기본적인 처리를 진행하였다. 또한 네트워크의 동기화를 빠르게 처리하기 위해서 임계 영역(critical section)을 적용했다.

5. 실험결과

5.1 인식 속도 실험

본 논문에서 제안한 모델을 활용하여 10종류의 제스처 인식 속도 실험을 진행하였다. 서버 측에는 C# 기반 서버 UI 및 C++ 기반의 DLL로 만들어진 Caffe [8] 라이브러리를 사용했다. 서버의 구현 상태는 Fig. 9와 같다. 게임 어플리케이션에 해당되는 클라이언트의 경우 유니티 5를 기반으로 작성했다. 본 실험에서는 제스처 패턴을 입력한 후 패킷화 하여 서버로 전송하는 과정을 수행한다. Windows 10, Ram 8GB, NVIDIA GTX 970m의 실험 환경을 구축했으며 제안한 CNN 모델과 전처리 과정을 포함하여 실험을 진행했다. 또한 인식 속도 실험을 위해서 총 100번의 입력을 시도 했으며 각 제스처 인식 시간을 측정했다. 실험 결과는 Fig. 10과 같으며 가로 축은 제스처를 입력한 시도 횟수(number of attempts), 세로 축

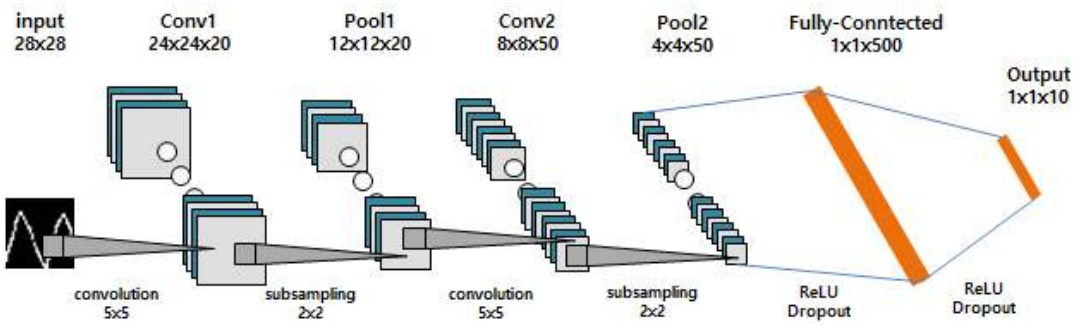


Fig. 7. Proposed model.

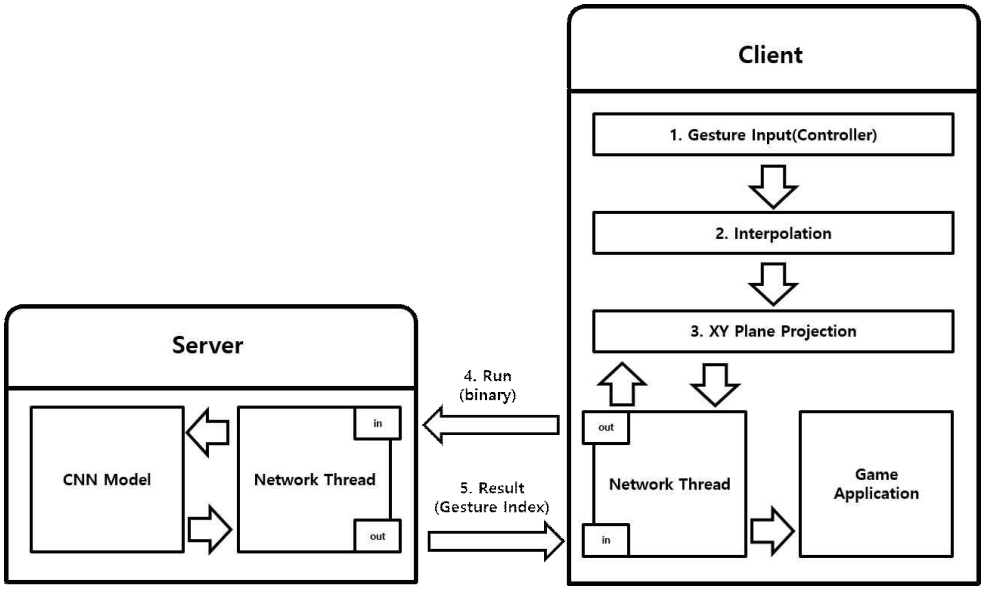


Fig. 8. Network communication structure for deep learning.

Fig. 9. Server program.

은 알고리즘이 실행된 시간을 나타낸다. 또한 실험 결과의 범례는 0에서 9까지 총 10개의 제스처를 표시했다.

본 실험에서는 클라이언트와 서버간의 송수신 과정에서 일어나는 지연 속도(latency)로 인한 오차를 줄이기 위해서 하나의 컴퓨터에서 네트워크 통신을 진행했다. 하지만 게임을 실행 시키는 메인 쓰레드와 CNN 모델에 대한 분류 결과를 처리하고 서버와 클라이언트 간의 통신을 처리하는 서브 쓰레드간의 연산 문제로 제스처를 입력할 때마다 소요하는 시간이 달라지는 것을 볼 수 있다. 2번 제스처에서 최대 18.1ms, 7번 제스처에서 최소 15.3ms의 소요 시간을 소모했다. 이외 제스처들의 소요 시간은 16.5ms에 가깝게 수렴하고 있다. 직접적으로 게임을 실시간으로 처리하는데 큰 무리가 없으며 최대 소요 시간 역시 크지가 않다.

5.2 분류 실험

모델을 활용하여 10종류의 제스처 분류 실험을 진행하였다. 학습한 데이터의 개수는 총 9000개 이며 정확도 측정을 위한 테스트 데이터는 분류별 100개씩 총 1000개이다. 학습 데이터의 경우 분류별 분포가 비슷하도록 구성하였다. 아래 Table 1은 본 논문에서 제시한 모델에 대한 실험결과로 평균적으로 99% 이상의 인식률을 나타낸다.

모델의 설정 값 중 학습률(learning rate)를 0.01로 설정하는 것이 과적합(over fitting)과 국부최소(local minima)[9]에 빠지는 것을 비교적 방지해주는 것으로 보인다. 평균값을 빼는 경우 보다 셔플(shuffle)과 그레이스케일(grayscale)을 적용 하는 것이 조금 더 좋은 효과를 나타냈다. 하지만 큰 차이는 나지 않았으며 매 실험마다 근소한 오차 값이 있으나 대부분 99% 이상의 인식률을 나타낸다.

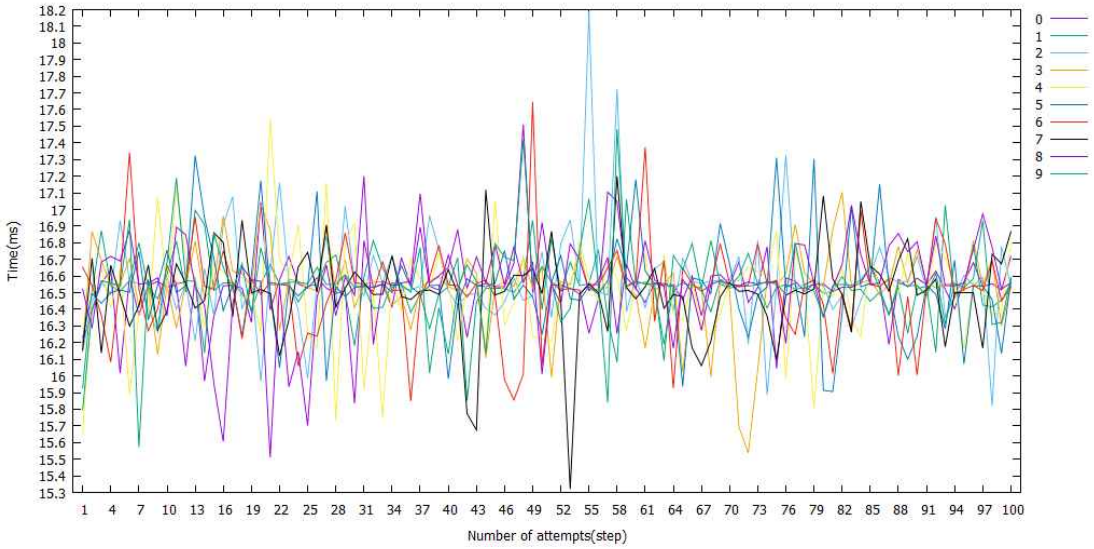


Fig. 10. Recognition time result of proposed method.

Table 1. Result of gesture classification

No	Learn Rate	Iteration	Additional Condition	Acc
1	0.01	20000	Shuffle	99.8%
2	0.01	20000	Shuffle, Mean	99.6%
3	0.01	20000	Shuffle, Grayscale	99.8%
4	0.01	20000	Shuffle, Mean, Grayscale	99.8%
5	0.001	20000	Shuffle	99.6%
6	0.001	20000	Shuffle, Mean	99.8%
7	0.001	20000	Shuffle, Grayscale	99.8%
8	0.001	20000	Shuffle, Mean, Grayscale	99.4%

기존 연구에서 활용했던 K-nearest neighbor 알고리즘에 기반한 $P recognizer[10]와 본 논문에서 제시한 모델의 인식률을 비교하는 실험을 진행했다. 실험에 활용한 모델의 구조는 Fig. 7과 동일하다. 또한 학습된 모델은 Table 2의 3번을 활용했다. 각 제스처 종류마다 100회를 입력 후 분류기에서 나온 인식률이다. 실험 결과는 Table 3과 같다.

Table 2를 보면 본 논문에서 제시한 모델은 평균적으로 98%의 인식률을 나타내고 있다. $P recognizer가 평균적으로 약 90%의 인식률을 보여준 것과 비교했을 때 상대적으로 높은 인식률을 기록했다. 각 제스처의 패턴의 특징을 살펴보았을 때, 입력하기 쉬운 2번, 4번과 달리 비교적 어려운 3번과 6번 패턴은 인식률의 차이가 크다. 하지만 패턴이 복잡한 7번은 인식률이 거의 동일하다. 패턴의 특징이 복잡할수록 다른 패턴들과 유사성이 거의 없어지기 때문에, 본 논문에서 제시한 모델과 $P recognizer 모두 인식률이 높게 측정된 것으로 예상된다.

Table 2. Comparison with $P recognizer

Samples	Proposed Method	$P recognizer
0	96%	91%
1	99%	84%
2	100%	100%
3	100%	66%
4	100%	99%
5	99%	99%
6	97%	82%
7	100%	99%
8	98%	88%
9	100%	100%

5.3 게임 어플리케이션 적용

제스처 인식을 실제적으로 활용하기 위해서 게임 콘텐츠에 적용했다. 특히나 가상현실 게임은 인터페이스에 대한 직관성이 떨어지기 때문에, 사용자가 게임에 적응하기 쉽지 않다. 이를 극복하기 위해서 제스처 인식과 같이 사용자가 쉽게 몰입할 수 있는 게임을 개발했다. 전체적인 게임 콘텐츠의 모습은 Fig. 11과 같다.

본 논문에서 제시한 CNN 모델을 활용하여 가상현실 게임의 콘텐츠에 적용했다. 적용된 게임은 Fig. 12와 같다. HTC VIVE의 컨트롤러를 활용하여 화면에서 출력되는 제스처의 패턴을 입력하고 맞추는 형식의 게임이다. 인지 재활을 목적으로 활용되었으며 특히 기억력에 해당하는 인지 게임의 종류로 사용됐다[11].

6. 결 론

본 논문에서는 2D 제스처를 분류하기 위해 CNN 모델 중 LeNet을 활용하여 연구를 진행하였다. 결과적으로 제스처 인식에 있어 99%가 넘는 정확도를 확인할 수 있었으며 기존의 알고리즘에 비해 높은 활용성을 보였다. 또한 게임 어플리케이션에 적용하기 위해 네트워크 구조를 기반으로 한 모델을 적용하여 제스처를 인식하는데 16.5ms의 수행 시간이 걸리며, 비교적 빠른 인식속도 통해서 실시간으로 제스처를 인식할 수 있었다. 이를 통해서 상용 게임에 대한 딥 러닝 모델의 활용 가능성을 확인하였다. 또한 가상현실 게임 콘텐츠에 적용하여 제스처 인식의 필요성에 대한 타당성을 확인하였으며, 추후 제스처 종류를 늘리고 평면에 투영함으로서 제한되는 공간의 활용을 극복 할 수 있는 방법에 대한 연구가 필요하다.

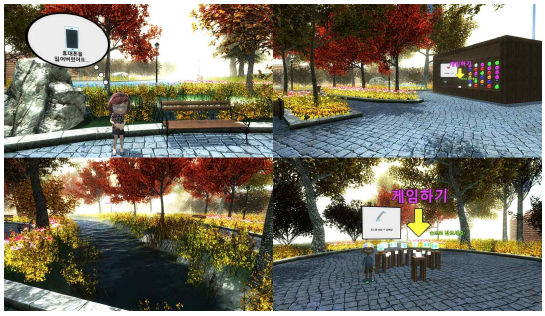


Fig. 11. VR game contents[11].

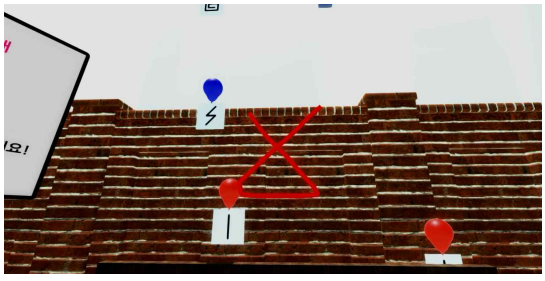


Fig. 12. Gesture recognition game[11]

REFERENCE

[1] H. Cheng, L. Yang, and Z. Liu, "Survey on 3D Hand Gesture Recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 26, No. 9, pp. 1659-1673, 2016.

[2] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks*, Vol. 61, pp. 85-117, 2015.

[3] J.H. Chae, J.H. Lim, and J.J. Lee "Gesture Classification Based on k-Nearest Neighbors Algorithm for Game Interface," *Journal of Korea Multimedia Society*, Vol. 19, No. 5, pp. 874-880, 2016.

[4] B. Fasel and D. Gatica-Perez, "Rotation-Invariant Neoperceptron," *Proceeding of 18th International Conference on Pattern Recognition*, pp. 336-339, 2006.

[5] R. McCartney, J. Yuan, and H.P. Bischof, "Gesture Recognition with the Leap Motion Controller," *Proceeding of International Conference on Image Processing, Computer Vision, and Pattern Recognition*, pp. 3-9, 2015.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278-2324, 1998.

[7] G.E. Dahl, T.N. Sainath, and G.E. Hinton, "Improving Deep Neural Networks for LVCSR Using Rectified Linear Units and Dropout," *Proceeding of IEEE International Conference on Acoustics, Speech and Signal*

Processing, pp. 8609-8613, 2013.

[8] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, et. al, "Caffe: Convolutional Architecture for Fast Feature Embedding," *Proceeding of ACM International Conference on Multimedia*, pp. 675-678, 2014.

[9] M. Gori and A. Tesi, "On the Problem of Local Minima in Backpropagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 1, pp. 76-86, 1992.

[10] R.D. Vatavu, L. Anthony, and J.O. Wobbrock, "Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes," *Proceeding of ACM International Conference on Multimodal Interaction*, pp. 273-280, 2012.

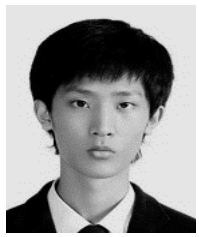
[11] J.H. Lim, J.H. Chae, H.S. Kim, H.K. Park, and J.J. Lee, "Study on VR Game for Rehabilitation Using Realtime Gesture Recognition," *Proceeding of Korea Information and Communication Society Winter Conference*, pp. 534-535, 2017.

채 지 훈

2016년 계명대학교 컴퓨터공학부 게임모바일공학전공 졸업(학사)
2016년~현재 계명대학교 컴퓨터공학부 석사과정
관심분야: 영상처리, 인공지능, 게임

임 종 헌

2013년 계명대학교 게임모바일콘텐츠학과 졸업(학사)
2015년 계명대학교 미디어아트학과 졸업(석사)
2015년~현재 계명대학교 컴퓨터공학부 박사과정
2012년~현재 rbxsoft 대표
관심분야: 영상처리, 인공지능, 게임



김 해 성

2011년~현재 계명대학교 컴퓨터공학부 게임모바일 전공 학사과정
관심분야: 그래픽스, 인공지능, 게임

이 준 재

1986년 경북대학교 전자공학과 졸업(학사)
1990년 경북대학교 전자공학과 졸업(석사)
1994년 경북대학교 전자공학과 졸업(박사)
1995년~2007년 동서대학교 컴퓨터 정보공학부 부교수
1998년~1999년 Georgia Institute of Technology 방문교수
2000년~2001년 (주)파미 연구소장
2007년~현재 계명대학교 컴퓨터공학부 교수
관심분야: 영상처리, 3D 컴퓨터 비전, 딥 러닝, 게임